最小生成树

2018年12月4日 15:17

-
- 最小生成树

一. 最小生成树Minimum Spanning Tree

1. 由原图全部n个顶点和其中n-1条边构成的无向树称为生成树
2. 边权和最小的成为最小生成树
3. 定理：最小生成树一定包含权最小的边
4. 推论：若想从生成森林中不包含的原图边中选出几条，将生成森林连成一棵生成树，则所选的边一定包含生成森林中不连通结点的权最小边

二. Kruskal ：维护最小生成森林

1. 初值：包含零条边的生产森林，即每个点自成一树
2. 循环：选出森林中不包含的边中权最小的，加入生成森林
3. 终点：选完了使n个结点成为生成树的n-1条边
4. 时间复杂度：O（mlogm）
5. 适用于存在权值相等的边的情况
6. int v,e; //点数、边数

```
int ans; //生成树的边权和
int rt[MN];

struct ST{
	int x,y,w;
	bool operator<(ST r){
		return w<r.w;   //让权小的排在前面
	}
}edge[MN*MN];

int get(int x){
	if(rt[x]==x)
		return x;
	return rt[x]=get(rt[x]);
}
int main(){
	cin>>v>>e;
	for__(i,1,v) rt[i]=i;
	for__(i,1,e) cin>>edge[i].x>>edge[i].y>>edge[i].w;
	sort(edge+1,edge+e+1);//确保遍历到的边是最小权的
	for__(i,1,e){
		int rx=get(edge[i].x);
		int ry=get(edge[i].y);
		if(rx!=ry)          //是需要加的边
			ans+=edge[i].w,
			rt[rx]=ry; }
	cout<<ans;
	return 0;
}
```

三. Prim算法：维护最小生成树的一部分

1. 时间复杂度O（n方），用二叉堆可优化到O（mlogn）
2. 用于：稠密图。尤其是完全图的最小生成树
3. 思路：
   i. 最初仅确定1号节点属于最小生成树
   ii. 找：端点分别确定属于和不确定属于最小生成树的、权最小的边
   iii. 删除该边不确定属于树的端点，加入树；把权加入总和
      -
      - 例题

一. 树成生小最（USSTD2I）

1. 题：已知一棵树，求构造完全图，要求其唯一生成树为已知树，求边权最小和
2. 引：利用kruskal的思路，若能找到两个连通集中有另一条边与已知树边等长，则生成树不会是它，若能找到更短的，那生成树就不会是已知树了，因此两连通集内其他边的权必须比已知边的权更大
3.
```
int n;
int rt[MN],sz[MN];
ll ans;
pair<int, pair<int,int> >edge[MN];
//权为first，起点为second.first，终点为second.second
int fd(int x){
    if(rt[x]==x)
        return x;
    return rt[x]=fd(rt[x]);}
void merge(int z,int x,int y){      //x并到y，边权为z
    //if(fd(x)==fd(y))return; //本题没有这种情况……
    int rx=fd(x);
    int ry=fd(y);
    ans+=((ll)sz[rx]*sz[ry] -1) *(z+1);
    sz[ry]+=sz[rx];
    rt[rx]=ry;}
```
4.
```
for__(i,1,n)
        sz[i]=1,
        rt[i]=i;
    int x,y,z;
    for_(i,1,n)
        scanf("%d%d%d",&x,&y,&z),
        edge[i]=make_pair(z,make_pair(x,y));
    sort(edge+1,edge+n);
    for_(i,1,n)
        merge(edge[i].first,
            edge[i].second.first,
            edge[i].second.second);
```